FIG. 1

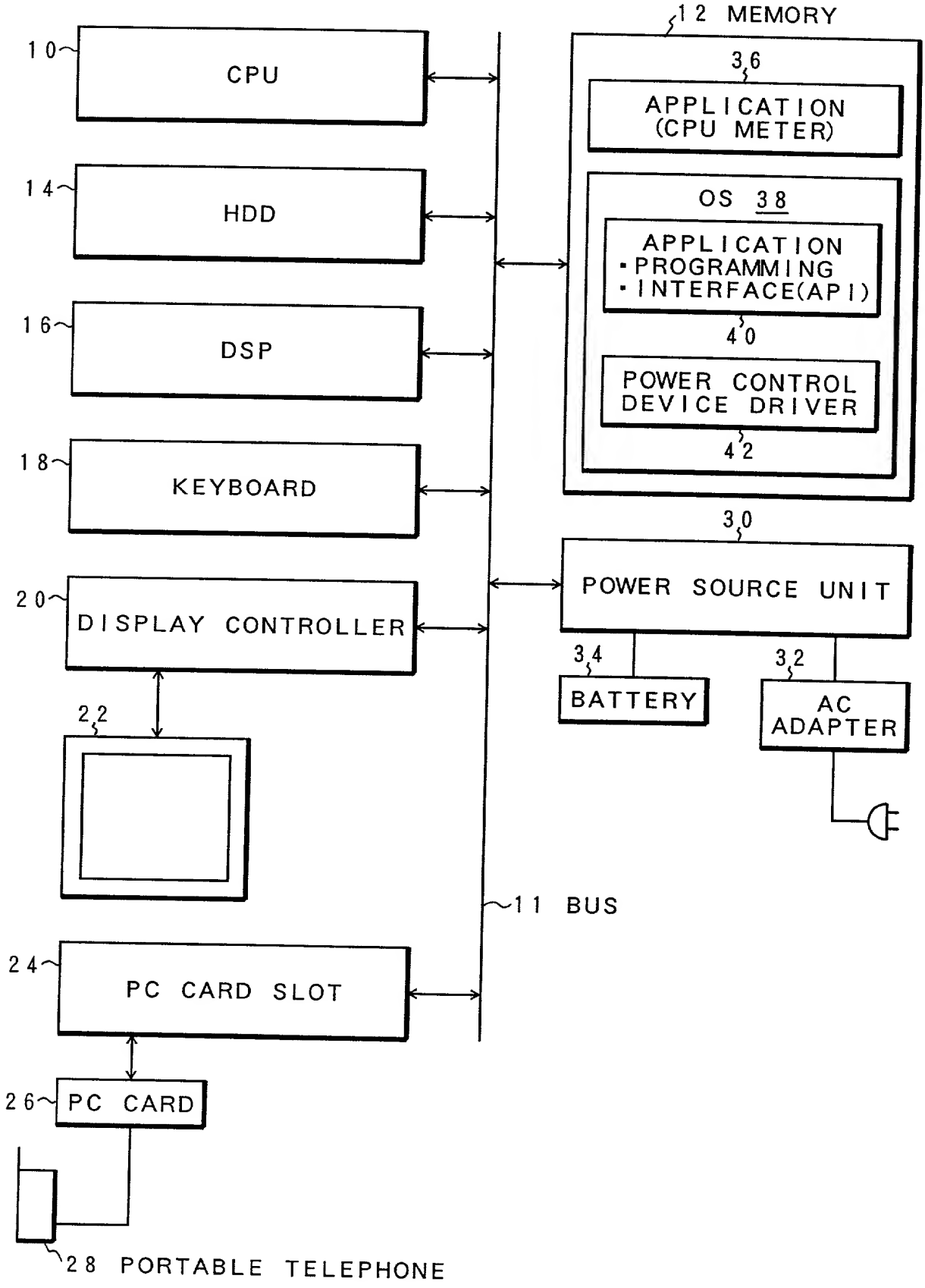

12 MEMORY
36
APPLICATION (CPU METER)
OS 38
APPLICATION
• PROGRAMMING
• INTERFACE(API)
40
POWER CONTROL DEVICE DRIVER
42
10
CPU
14
HDD
16
DSP
18
KEYBOARD
20
DISPLAY CONTROLLER
22
30
POWER SOURCE UNIT
34
BATTERY
32
AC ADAPTER
11 BUS
24
PC CARD SLOT
26
PC CARD
28 PORTABLE TELEPHONE

FIG. 2

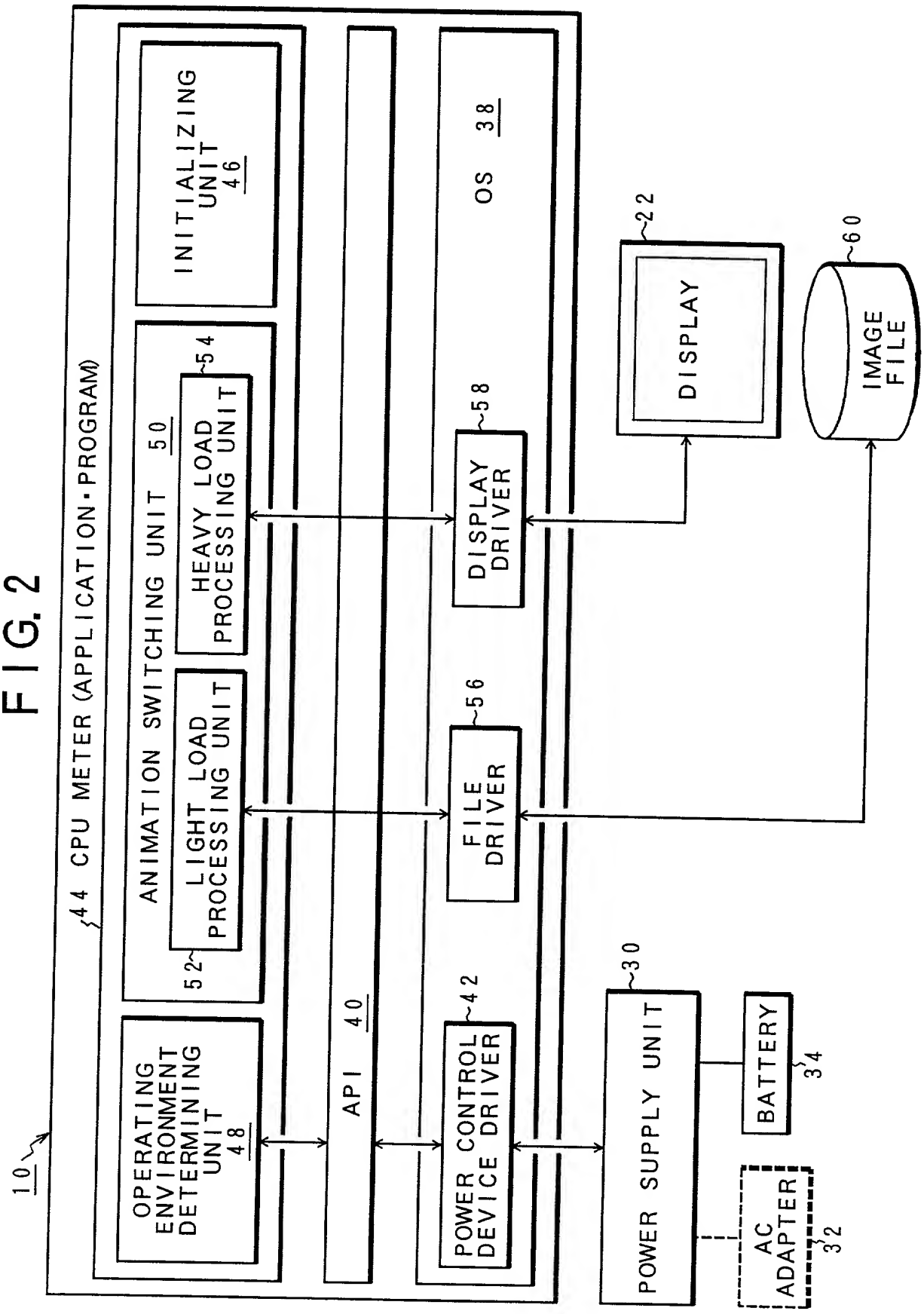

10
44 CPU METER (APPLICATION・PROGRAM)
OPERATING ENVIRONMENT DETERMINING UNIT 48
ANIMATION SWITCHING UNIT 50
LIGHT LOAD PROCESSING UNIT
52
HEAVY LOAD PROCESSING UNIT
54
INITIALIZING UNIT 46
API 40
POWER CONTROL DEVICE DRIVER
42
FILE DRIVER
56
DISPLAY DRIVER
58
OS 38
POWER SUPPLY UNIT
30
BATTERY
34
AC ADAPTER
32
DISPLAY
22
IMAGE FILE
60

FIG. 3A
BASIC IMAGES

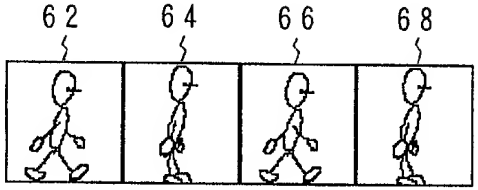


FIG. 3B
ANIMATION

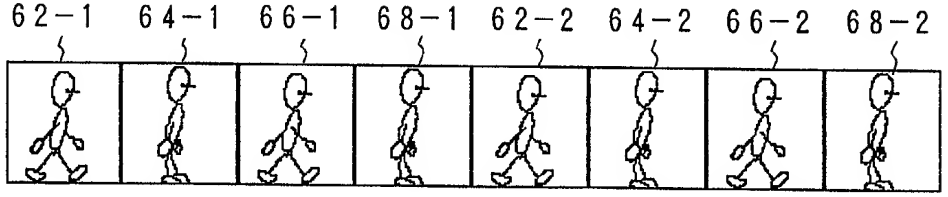


FIG. 3C
STILL IMAGE

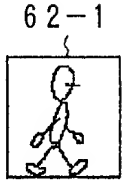


FIG. 4

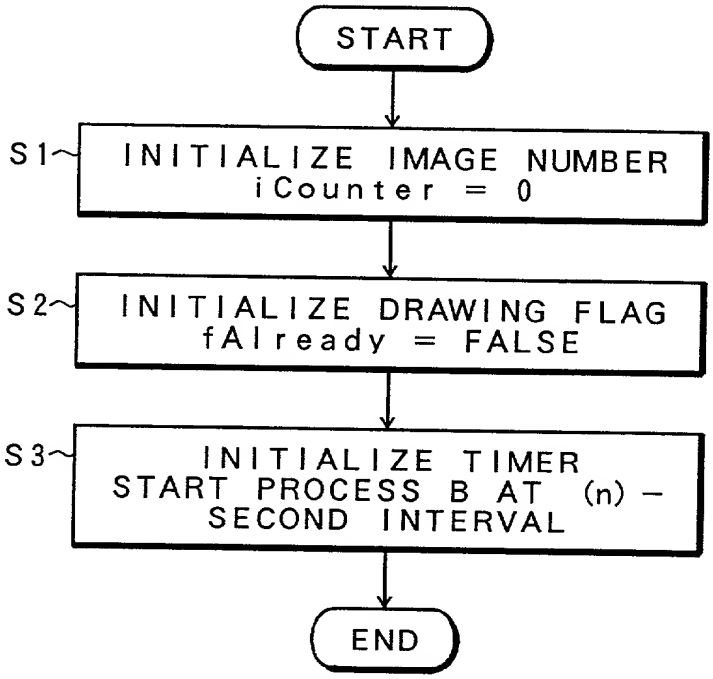

START
S1
INITIALIZE IMAGE NUMBER
iCounter = 0
S2
INITIALIZE DRAWING FLAG
fAlready = FALSE
S3
INITIALIZE TIMER
START PROCESS B AT (n) -
SECOND INTERVAL
END

FIG. 5

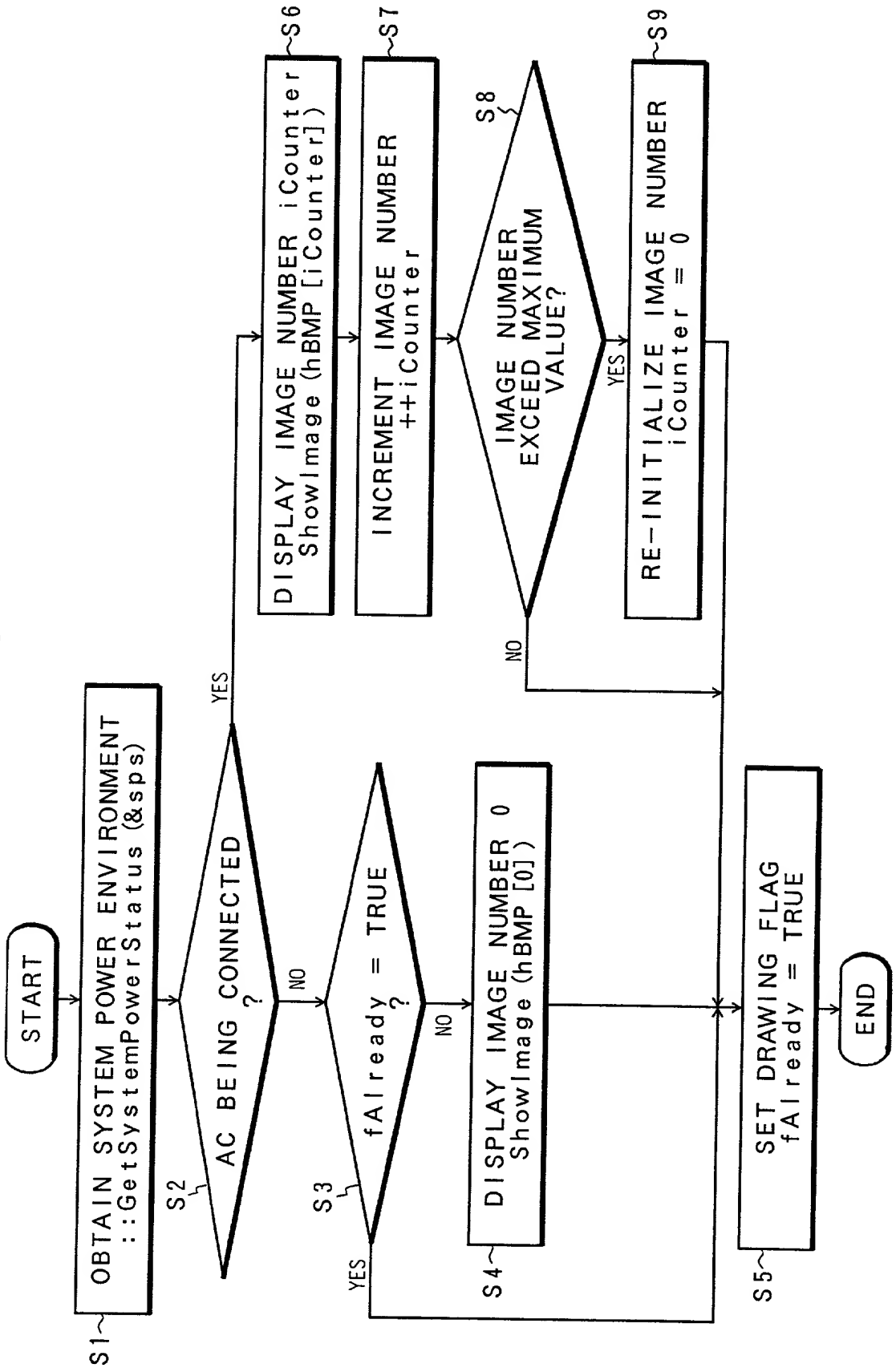
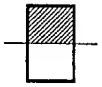

START
OBTAIN SYSTEM POWER ENVIRONMENT
::GetSystemPowerStatus (&sps)
S1
AC BEING CONNECTED ?
S2
YES
NO
fAlready = TRUE ?
S3
YES
NO
DISPLAY IMAGE NUMBER 0
ShowImage (hBMP [0])
S4
SET DRAWING FLAG
fAlready = TRUE
S5
END
DISPLAY IMAGE NUMBER iCounter
ShowImage (hBMP [iCounter])
S6
INCREMENT IMAGE NUMBER
++iCounter
S7
IMAGE NUMBER
EXCEED MAXIMUM
VALUE?
S8
YES
NO
RE-INITIALIZE IMAGE NUMBER
iCounter = 0
S9

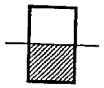
FIG. 6A



List.1

```
void sample()
{
        //-------------------------------------------------------------------------
        // #a  SYSTEM POWER STATUS STORAGE AREA
        //-------------------------------------------------------------------------
        SYSTEM_POWER_STATUS sps;
        //-------------------------------------------------------------------------
        // #b  OBTAIN SYSTEM POWER STATUS
        //-------------------------------------------------------------------------
        ::GetSystemPowerStatus( &sps );
        //-------------------------------------------------------------------------
        // #c  DRAWING FLAG WHICH IS SET(INITIALIZED ONLY ONCE)
        //-------------------------------------------------------------------------
        static BOOL  fAlready = FALSE;
        //-------------------------------------------------------------------------
        // #d  SYSTEM BEING OPERATED ON BATTERY?
        //-------------------------------------------------------------------------
        if ( AC_LINE_OFFLINE! = sps.ACLineStatus )  {
                //-----------------------------------------------------------------
                // #e1  NO IMAGE DRAWN
                //-----------------------------------------------------------------
                if ( !fAlready )   {
                        //---------------------------------------------------------
                        // #e2  DRAW IMAGE
                        //---------------------------------------------------------
                        ShowImage( hBMP[0] );
                //-----------------------------------------------------------------
                // #e3  ALREADY DRAWN
                //-----------------------------------------------------------------
                }  else  {
```

FIG. 6B



```
                    //-----------------------------------------------------------
                    // #e4  NO OPERATION
                    //-----------------------------------------------------------
            }
    //-------------------------------------------------------------------------
    // #f1  SYSTEM BEING OPERATED ON AC ADAPTER
    //-------------------------------------------------------------------------
    }  else  {
            //-----------------------------------------------------------
            // #f2  ANIMATION COUNTER(INITIALIZED ONLY ONCE)
            //-----------------------------------------------------------
            static int iCounter = 0;
            //-----------------------------------------------------------
            // #f3  DRAW IMAGE
            //-----------------------------------------------------------
            ShowImage( hBMP[i] );
            //-----------------------------------------------------------
            // #f4  INCREMENT ANIMATION COUNTER
            //-----------------------------------------------------------
            if ( ANIMATION_MAX < ++iCounter )  {
                    //-----------------------------------------------------------
                    // #f5  IF LAST IMAGE HAS BEEN DRAWN, RETURN TO THE FIRST
                    //-----------------------------------------------------------
                    iCounter = 0;
            }
    }
    //-------------------------------------------------------------------------
    // #g  ALREADY DRAWN
    //-------------------------------------------------------------------------
    fAlready = TRUE;

    return;
```

FIG. 7A

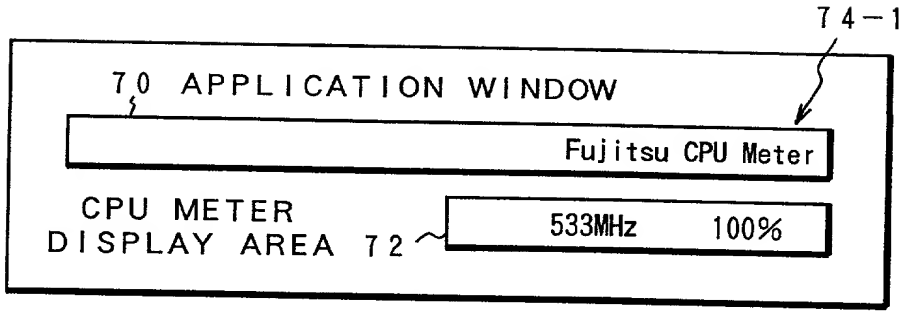


FIG. 7B

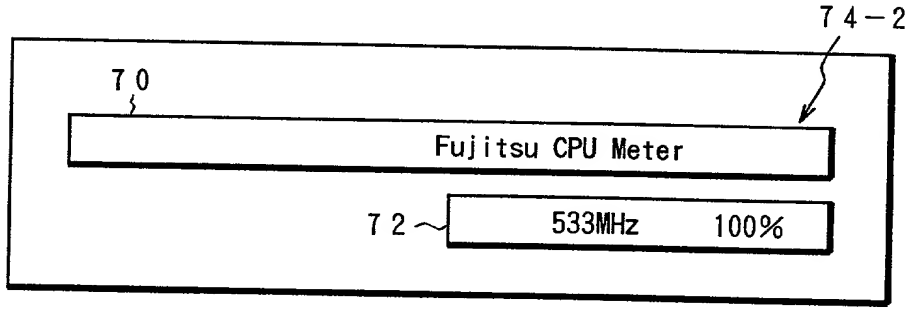


FIG. 7C

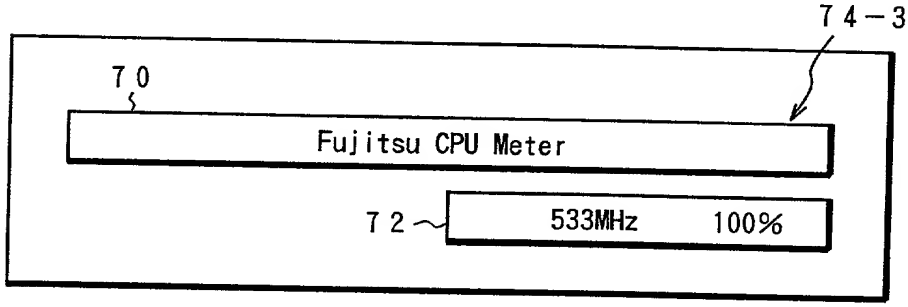


FIG. 7D

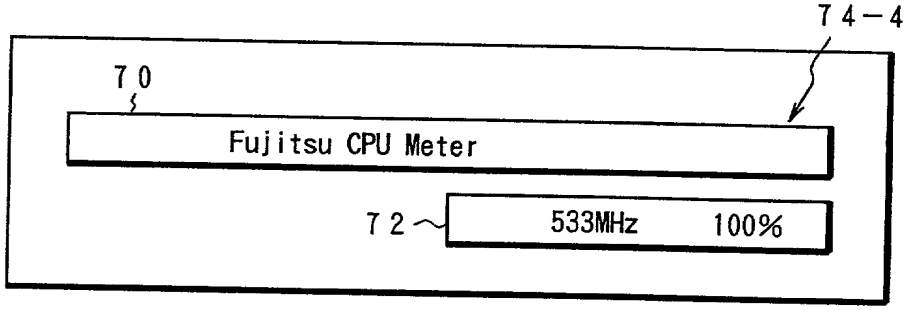


FIG. 7E

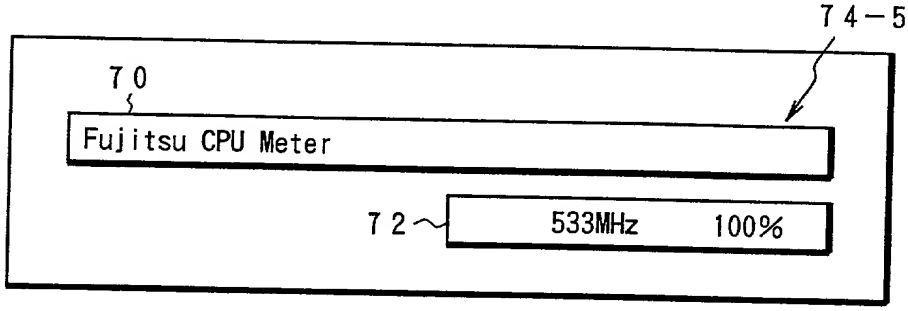
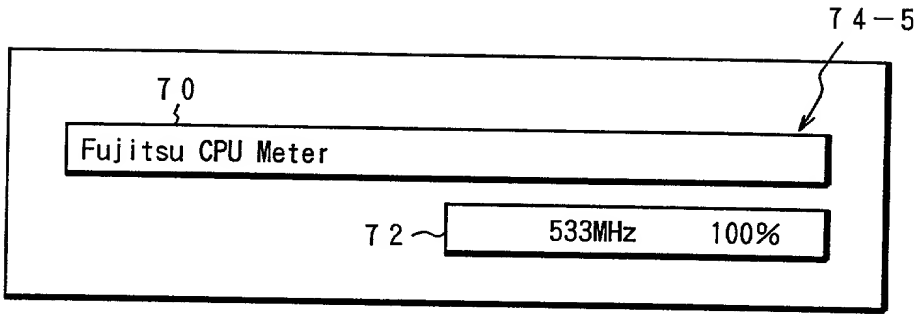


FIG. 8



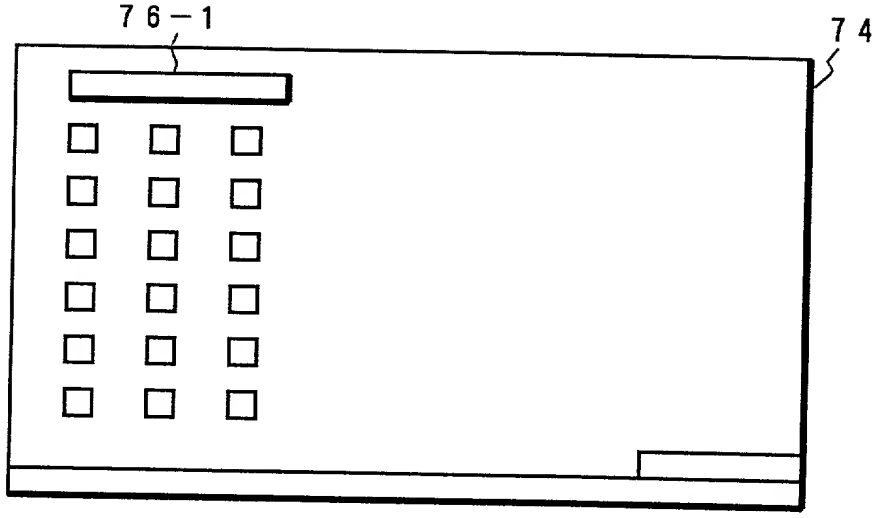


FIG. 9A

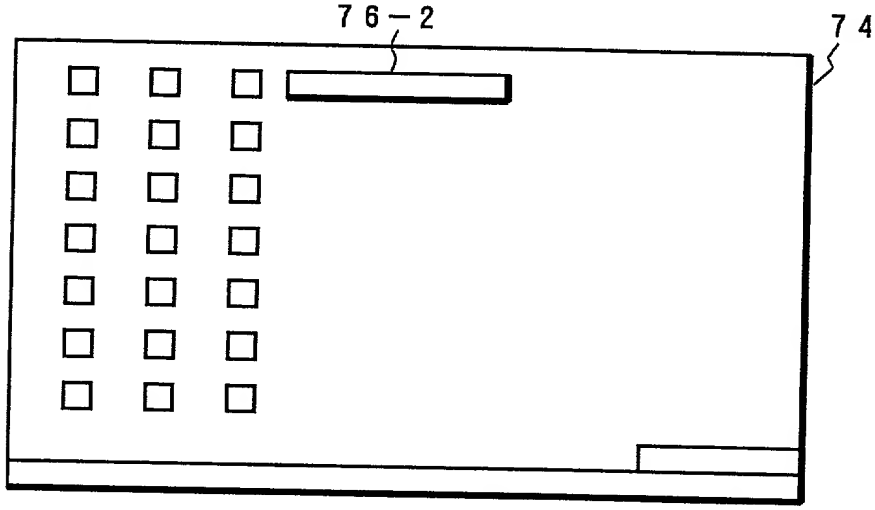


FIG. 9B

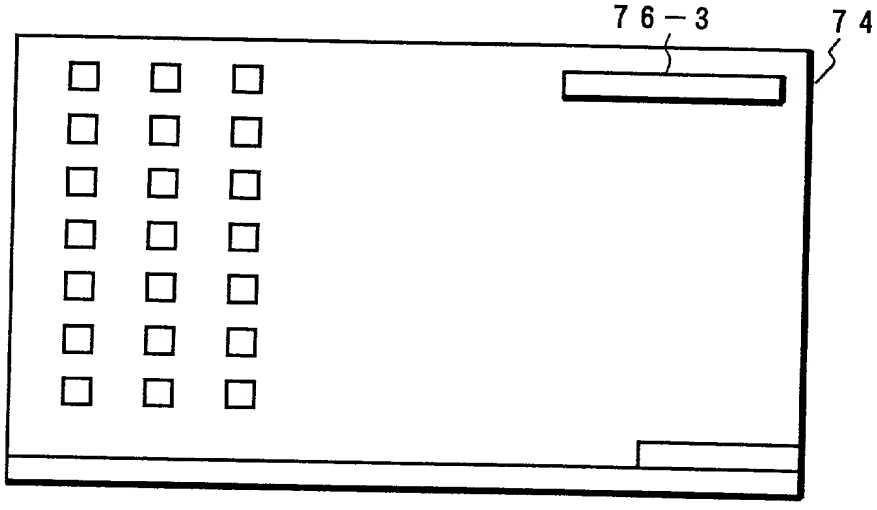


FIG. 9C

FIG. 10

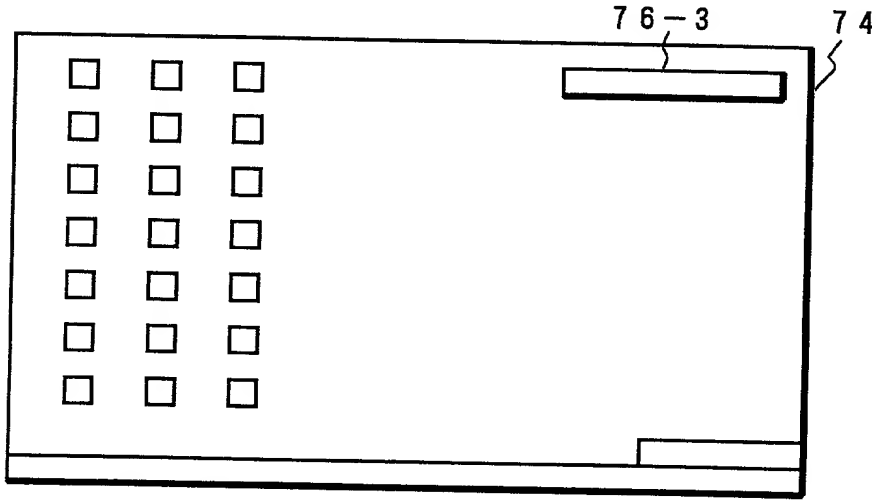


FIG. 11

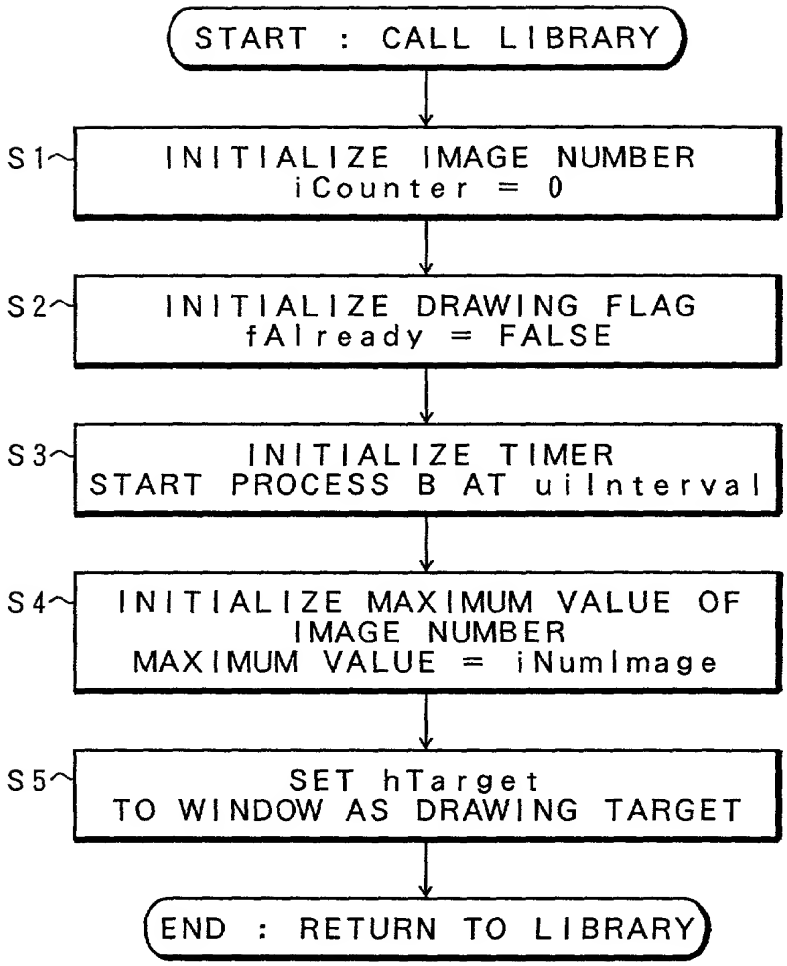

START : CALL LIBRARY
S1
INITIALIZE IMAGE NUMBER
iCounter = 0
S2
INITIALIZE DRAWING FLAG
fAlready = FALSE
S3
INITIALIZE TIMER
START PROCESS B AT uiInterval
S4
INITIALIZE MAXIMUM VALUE OF IMAGE NUMBER
MAXIMUM VALUE = iNumImage
S5
SET hTarget
TO WINDOW AS DRAWING TARGET
END : RETURN TO LIBRARY

FIG. 12

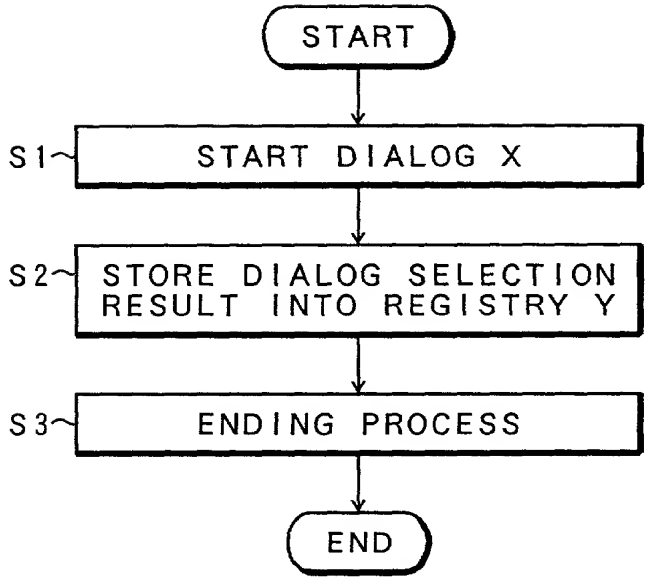

START
S1
START DIALOG X
S2
STORE DIALOG SELECTION RESULT INTO REGISTRY Y
S3
ENDING PROCESS
END

FIG. 13

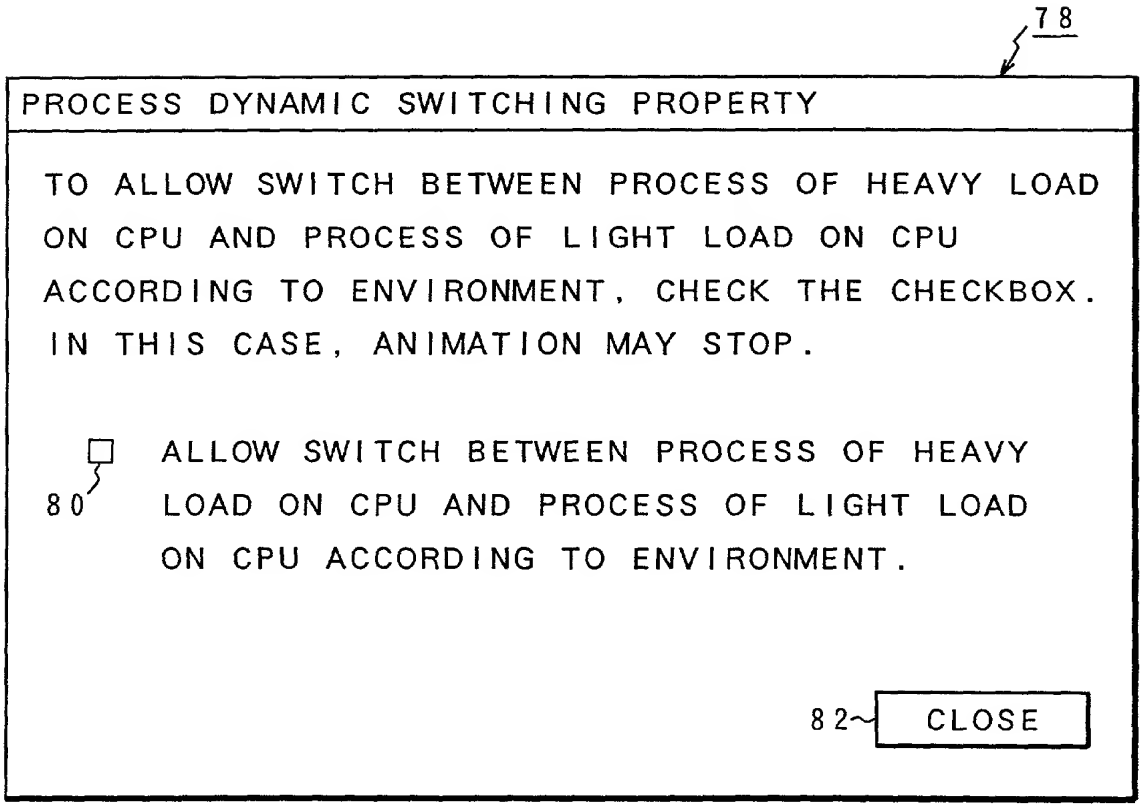
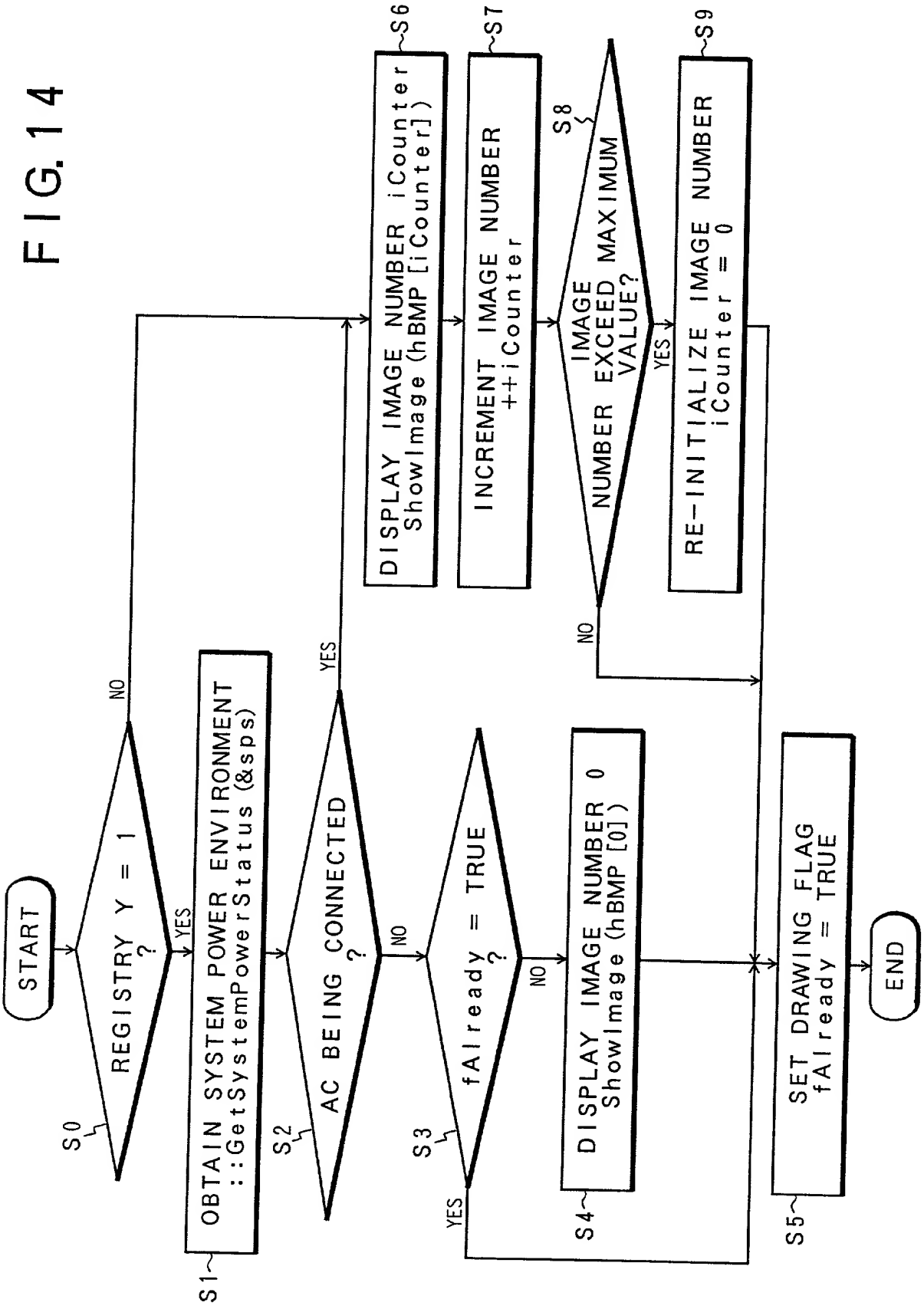

78
PROCESS DYNAMIC SWITCHING PROPERTY
TO ALLOW SWITCH BETWEEN PROCESS OF HEAVY LOAD ON CPU AND PROCESS OF LIGHT LOAD ON CPU ACCORDING TO ENVIRONMENT, CHECK THE CHECKBOX. IN THIS CASE, ANIMATION MAY STOP.
80
ALLOW SWITCH BETWEEN PROCESS OF HEAVY LOAD ON CPU AND PROCESS OF LIGHT LOAD ON CPU ACCORDING TO ENVIRONMENT.
82
CLOSE

FIG.14


START
S0
REGISTRY Y = 1 ?
NO
YES
S1
OBTAIN SYSTEM POWER ENVIRONMENT
::GetSystemPowerStatus (&sps)
S2
AC BEING CONNECTED ?
YES
NO
S3
fAlready = TRUE ?
YES
NO
S4
DISPLAY IMAGE NUMBER 0
ShowImage (hBMP [0])
S5
SET DRAWING FLAG
fAlready = TRUE
END
S6
DISPLAY IMAGE NUMBER iCounter
ShowImage (hBMP [iCounter])
S7
INCREMENT IMAGE NUMBER
++iCounter
S8
IMAGE NUMBER EXCEED MAXIMUM VALUE?
YES
NO
S9
RE-INITIALIZE IMAGE NUMBER
iCounter = 0